Gradient Descent

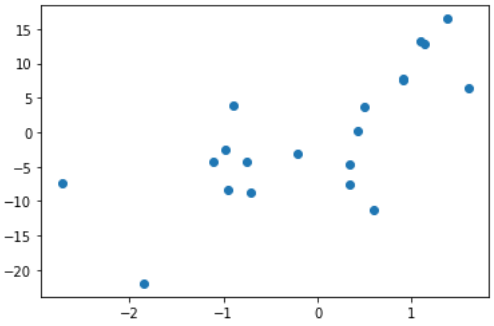
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn.datasets import make_regression
```

In [3]:
```
x,y=make_regression(n_samples=20,n_features=1,noise=6)
```

In [4]:
```
plt.scatter(x,y)
```

Out[4]:
```
<matplotlib.collections.PathCollection at 0x20d53078f70>
```



In [5]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
```

In [6]:
```
lr=LinearRegression()
```

In [7]:
```
lr.fit(x,y)
```

Out[7]:
```
LinearRegression()
```

In [8]:
```
m=lr.coef_
m
```

Out[8]:
```
array([5.85286116])
```

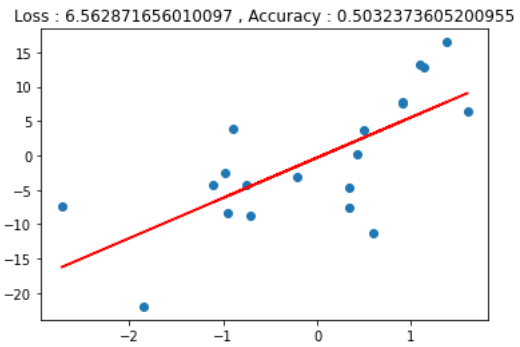
In [9]:
```
b=lr.intercept_
b
```

Out[9]:
```
-0.3158894920795396
```

In [10]:
```
),'r-')

sqrt(mean_squared_error(y,lr.predict(x)))} , Accuracy : {r2_score(y,lr.predict(x))}') # f input pass karne ke liye variable print
```

Out[10]:
```
Text(0.5, 1.0, 'Loss : 6.562871656010097 , Accuracy : 0.5032373605200955')
```



In [11]:
```
x
```

Out[11]:
```
array([[-0.89951499],
       [ 0.42677376],
       [ 0.91414247],
       [-0.21438714],
       [ 0.49327658],
       [ 1.37591529],
       [ 1.14027021],
       [-0.97723461],
       [ 0.333318  ],
       [-0.95399285],
       [-0.75458818],
       [-1.11263508],
       [ 0.91304615],
       [-0.71439226],
       [ 0.34447633],
       [-1.84803865],
       [-2.72307129],
       [ 1.09538405],
       [ 1.60421963],
       [ 0.60056921]])
```

In [12]:
```
x.ravel() # 1 d convert
```

Out[12]:
```
array([-0.89951499,  0.42677376,  0.91414247, -0.21438714,  0.49327658,
        1.37591529,  1.14027021, -0.97723461,  0.333318  , -0.95399285,
       -0.75458818, -1.11263508,  0.91304615, -0.71439226,  0.34447633,
       -1.84803865, -2.72307129,  1.09538405,  1.60421963,  0.60056921])
```

In [13]:
```
x1=x
y1=y
```

In [14]:
```
class GDRegressor:

    def __init__(self,learning_rate,epochs):
        self.m=0
        self.b=0
        self.lr=learning_rate
        self.epochs=epochs

    def fit(self,x,y):
        # calculate the b using GD
        for i in range(self.epochs):
            loss_slop_b=-2*np.sum(y-self.m*x.ravel()-self.b)
            loss_slop_m=-2*np.sum((y-self.m*x.ravel()-self.b)*x.ravel())

            self.b=self.b-(self.lr*loss_slop_b)
            self.m=self.m-(self.lr*loss_slop_m)

        print(self.m,self.b)

    def predict(self,x):
        return self.m*x+self.b
```

In [15]:
```
gd=GDRegressor(0.001,1000)
```

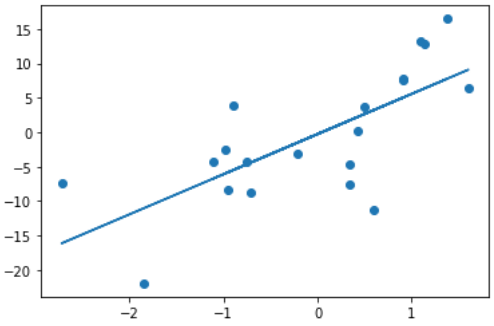
In [16]:
```
gd.fit(x,y) # value of m & b
```

```
5.852861159972977 -0.3158894920795407
```

In [17]:
```
m = 0
b =3
lr = 0.001
hh = []
slope = []
intercept = []
for i in range(100):
    loss_slope_b = -2 * np.sum(y - m*x.ravel() - b)
    loss_slope_m = -2 * np.sum((y - m*x.ravel() - b)*x.ravel())

    b = b - (lr * loss_slope_b)
    m = m - (lr * loss_slope_m)
    yhat= np.sqrt(mean_squared_error(y,(m*x)+b))
    ht = hh.append(yhat)
    ss = slope.append(m)
    ii= intercept.append(b)
    print(f"Slope {m}, yintercept {b}, Loss {yhat}")
    #print(hh)
plt.plot(x,slope[i] *x + intercept[i])
plt.scatter(x,y)
plt.show()
```

```
Slope 5.822920910738001, yintercept -0.2649402367623293, Loss 6.56316767589662
Slope 5.824546536479782, yintercept -0.2670354804789334, Loss 6.563141503469027
Slope 5.826085181517762, yintercept -0.26904380474389267, Loss 6.563117681757677
Slope 5.827541451656034, yintercept -0.27096885272298865, Loss 6.563095996401981
```



In [ ]: